

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **07244595 A**(43) Date of publication of application: **19.09.95**

(51) Int. Cl.

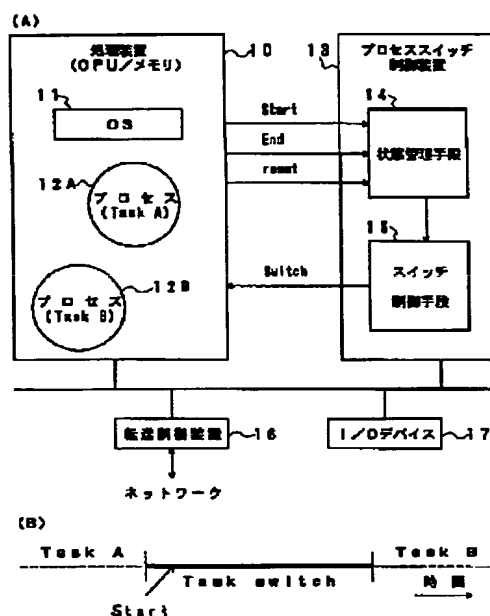
G06F 9/46(21) Application number: **06035535**(22) Date of filing: **07.03.94**(71) Applicant: **FUJITSU LTD**(72) Inventor: **SHIMIZU TOSHIYUKI
ISHIHATA HIROAKI****(54) PROCESS SWITCH CONTROLLER AND
PROCESS CONTROLLING METHOD**

(57) Abstract:

PURPOSE: To attain highly efficient message transmission and I/O without passing it through an OS by controlling a process switch signal so that a process switch is not turned on in a switch suppressing state.

CONSTITUTION: A state managing means 14 in a process switch controller 13 manages a state including at least a free state capable of turning on the process switch and a switch suppressing state for suppressing the turning-on of the process switch because a process being executed at present is in input/output processing or in message communication. A switch control means 15 provides timing for turning on the process switch with width in accordance with a state, and in the switch suppressing state, controls a process switch signal so as not to turn on the switch. Thereby the sending or I/O of a message can be directly controlled in a user level even in a multi-processing environment, the control can be attained by low overhead and the performance of a computer can be efficiently applied.

COPYRIGHT: (C)1995,JPO



This Page Blank (uspto)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-244595

(43) 公開日 平成7年(1995)9月19日

(51) Int.Cl.⁶

G 0 6 F 9/46

識別記号

3 4 0 B 7737-5B

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数7 O L (全 11 頁)

(21) 出願番号 特願平6-35535

(22) 出願日 平成6年(1994)3月7日

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中1015番地

(72) 発明者 清水 俊幸

神奈川県川崎市中原区上小田中1015番地

富士通株式会社内

(72) 発明者 石畑 宏明

神奈川県川崎市中原区上小田中1015番地

富士通株式会社内

(74) 代理人 弁理士 小笠原 吉義 (外2名)

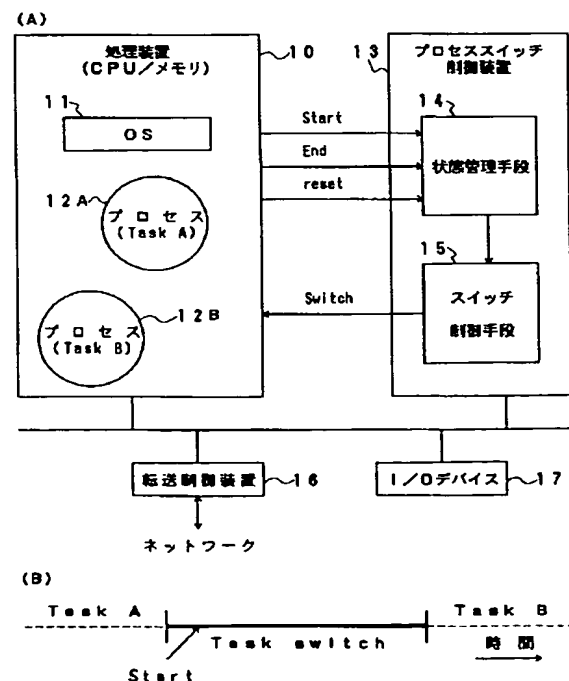
(54) 【発明の名称】 プロセススイッチ制御装置およびプロセス制御方法

(57) 【要約】

【目的】 マルチプロセッシング・システムにおけるプロセススイッチ制御装置およびプロセス制御方法に関し、マルチプロセッシング環境でのユーザレベルの入出力およびメッセージ通信を可能とし、低いオーバーヘッドの入出力およびメッセージ通信を実現することを目的とする。

【構成】 プロセススイッチを起こすことのできるフリー状態とクリティカルな状態とを状態管理手段14により管理し、これらの状態に応じてプロセススイッチを起こすタイミングに幅を持たせ、現在実行中のプロセスが入出力中またはメッセージ通信中のクリティカルな状態にある場合には、スイッチ制御手段15によりプロセススイッチが起きないようにプロセスのスイッチ信号を制御する。

本発明の原理説明図



【特許請求の範囲】

【請求項 1】 複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおいて用いられるプロセススイッチ制御装置であって、少なくとも、プロセススイッチを起こすことのできるフリー状態と、現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態とを含む状態を管理する状態管理手段(14)と、前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ、前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御するスイッチ制御手段(15)とを備えたことを特徴とするプロセススイッチ制御装置。

【請求項 2】 複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおいて用いられるプロセススイッチ制御装置であって、少なくとも、プロセススイッチを起こすことのできるフリー状態と、現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態と、プロセススイッチのトリガとなるタイマー・イベントが所定の時間内に発生する状態になったことを示すスイッチ予告促進状態とを含む状態を管理する状態管理手段(14)と、前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ、前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御するとともに、前記スイッチ予告促進状態において現在実行中のプロセスが入出力中またはメッセージ通信中に入ることを示すイベントが発生したときには、そのプロセスが入出力中またはメッセージ通信中に入る前にプロセスのスイッチ信号を出力するスイッチ制御手段(15)とを備えたことを特徴とするプロセススイッチ制御装置。

【請求項 3】 請求項 1 または請求項 2 記載のプロセススイッチ制御装置において、前記スイッチ制御手段(15)は、前記スイッチ抑止状態においてはプロセススイッチを遅延させ、現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベントによりプロセスのスイッチ信号を出力し、プロセススイッチを起こす手段を備えたことを特徴とするプロセススイッチ制御装置。

【請求項 4】 請求項 3 記載のプロセススイッチ制御装置において、前記スイッチ制御手段(15)は、前記スイッチ抑止状態におけるプロセススイッチの遅延が所定の時間を越えた場合には割り込みを起こし、一つのプロセスが実行権を占有しないように制御する手段を備えたことを特徴とするプロセススイッチ制御装置。

【請求項 5】 請求項 3 記載のプロセススイッチ制御装置において、前記スイッチ制御手段(15)は、前記スイッチ抑止状態において非同期的な外部割り込みがあった場合に、その割り込みの発生を遅延させ、現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベント

が発生してから保留した割り込みを発生させる手段を備えたことを特徴とするプロセススイッチ制御装置。

【請求項 6】 複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおけるプロセス制御方法において、入出力を実行するプロセス、またはメッセージを送信もしくは受信するプロセスを一つに制限するために、一つのプロセスが入出力を開始またはメッセージの送信もしくは受信を開始してからそれが終了するまでのクリティカルな状態を管理し、そのクリティカルな状態においては他の実行可能なプロセスが入出力を開始またはメッセージの送信もしくは受信を開始することを抑止し、入出力またはメッセージの送信もしくは受信を、マルチプロセッシング環境において複数のプロセスからユーザプログラムのレベルで起動可能とすることを特徴とするプロセス制御方法。

【請求項 7】 複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおけるプロセス制御方法において、入出力を実行するプロセスまたはメッセージを送信もしくは受信するプロセスを一つに制限するために、所定のアドレス空間のアクセス権を、入出力を開始したプロセスまたはメッセージの送信もしくは受信を開始したプロセスに与え、他の実行可能なプロセスの前記アドレス空間へのアクセスを禁止し、前記アクセス権を持つプロセスの入出力が終了またはメッセージの送信もしくは受信が終了したときに前記アクセス権を変更し、入出力またはメッセージの送信もしくは受信を、複数のプロセスからユーザプログラムのレベルで起動可能とすることを特徴とするプロセス制御方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおけるプロセススイッチ制御装置および制御方法に関するものである。

【0002】 近年、計算機システムの高速化が要求されている。このような計算機システムでは、一般に複数のプロセスが走行している。特に、計算性能を上げるために複数のシステムを接続して、相互に通信を行うシステムも用いられており、この場合に利用されるメッセージ通信は、計算性能を犠牲にしないためにもオーバーヘッドができるだけ小さいことが要求される。同様に入出力においてもオーバーヘッドが小さいことが要求される。なお、ここでプロセスとは CPU 実行権を得て走行する計算機システムの実行単位をいい、この実行単位にはいわゆるタスクと呼ばれるものも含まれる。

【0003】

【従来の技術】 従来、計算機システムにおいて複数のプロセスが走行している場合、一連の入出力(I/O)処理にはオペレーティング・システム(以下、OSという)を介在させる必要があった。しかし、OSを介在さ

せると、ユーザプロセスからOSへの制御移行に伴う処理などのために、非常に計算機の処理時間がかかる。このOSによるオーバーヘッドを削減するために、OSを介さないユーザレベルの転送方式として、特開平3-150659号公報（データ転送方式）、特開平5-233440号公報（バッファ機能を備えたデータ転送方式）などに示される技術が知られている。

【0004】しかし、従来のこれらの方式は、1プロセスあたりメッセージを送信できるプロセスが一つである場合に利用できる技術であり、マルチプロセッシング環境では利用するのは難しかった。

【0005】

【発明が解決しようとする課題】従来、OSを介さずにユーザレベルでメッセージ送信や、入出力操作を可能にしようとした場合、マルチプロセッシング環境では、一つの操作（例えばメッセージ送信）が複数のトランザクションに分割された場合に、トランザクションの間にプロセススイッチが発生して、他のプロセスによる操作が割り込み、一つの操作として意味を持たなくなってしまうという問題があった。

【0006】図12は本発明の課題説明図である。図12(A)は、一般のプロセススイッチを表している。タスクAが走行しているときに、プロセススイッチを起こす割り込みが入ると、OSがタスクBにCPU時間を与えてタスクBが走り出す。

【0007】図12(B)は、メッセージ送信や入出力をStartからEndまでタスクAが行っている様子を表している。ここで、タスクAが(1)に示すように入出力を開始する前に、プロセススイッチを起こす割り込みがあった場合には、プロセススイッチによってCPU実行権をタスクBに切り替えても何ら問題はない。これに対して、(2)に示すようにメッセージ送信や入出力を、OSを介さずにユーザレベルでタスクAが行っている最中(critical section)に、プロセススイッチを起こす割り込みが発生すると、問題が発生する。

【0008】メッセージ送信や入出力がユーザレベルで行われているために、ここで、プロセススイッチによりタスクBがCPU時間を与えられて走行を開始すると、タスクAと同様なメッセージ送信や入出力を行うことがあり、タスクAの操作と競合が発生して正しい操作が行われなくなる。

【0009】本発明は上記問題点の解決を図り、マルチプロセッシング環境においても、OSを介さない効率の高いメッセージの送信および入出力を可能とすることを目的とする。

【0010】

【課題を解決するための手段】図1は本発明の原理説明図である。図1(A)において、10はCPUおよびメモリなどからなる処理装置、11はマルチプロセッシング環境を提供するオペレーティング・システム(O

S)、12A、12BはCPU実行権を得て走行する計算機システムの実行単位であるプロセス（以下の説明では、プロセスの例としてタスクを代表させて説明する）、13はプロセスの切り替えを制御するプロセススイッチ制御装置、14はプロセスの切り替えを制御するための状態を管理する状態管理手段、15はプロセスを切り替えるタイミング信号であるスイッチ信号(Switch)の出力を制御するスイッチ制御手段、16はネットワーク等へのメッセージの転送を制御する転送制御装置、17はディスク装置等のI/Oデバイスを表す。

【0011】また、Startは、現在実行中のプロセスが転送制御装置16によるメッセージ通信またはI/Oデバイス17に対する入出力を開始するというイベントを示す信号、Endは、現在実行中のプロセスが転送制御装置16によるメッセージ通信またはI/Oデバイス17に対する入出力を終了するというイベントを示す信号、resetは、状態をクリアすることを指示する信号、Switchは、プロセススイッチを起こすことを指示する信号を表す。

20 【0012】処理装置10では、OS11の制御のもとに、複数のプロセス（タスクA）12A、プロセス（タスクB）12B、…が時分割的にCPU時間を与えられて走行する。本発明では、これらの複数のタスクA、B、…の走行を切り替えるために、状態管理手段14およびスイッチ制御手段15を持つプロセススイッチ制御装置13を有する。

30 【0013】請求項1記載の発明では、プロセススイッチ制御装置13の状態管理手段14は、少なくとも、プロセススイッチを起こすことのできるフリー状態と、現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態とを含む状態を管理する。また、スイッチ制御手段15は、前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ、前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御する。

40 【0014】請求項2記載の発明では、状態管理手段14は、少なくとも、プロセススイッチを起こすことのできるフリー状態と、現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態と、プロセススイッチのトリガとなるタイマー・イベントが所定の時間内に発生する状態になったことを示すスイッチ予告促進状態とを含む状態を管理する。また、スイッチ制御手段15は、前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ、前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御するとともに、前記スイッチ予告促進状態において現在実行中のプロセスが入出力中またはメッセージ通信中に入ることを示すイベントが発生したと

きには、そのプロセスが入出力中またはメッセージ通信中に入る前にプロセスのスイッチ信号を出力する。

【0015】請求項3記載の発明では、さらにスイッチ制御手段15は、前記スイッチ抑止状態においてプロセススイッチを起こすためのタイマー・イベントまたはその予告イベントが発生した場合に、プロセススイッチを遅延させ、現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベントによりプロセスのスイッチ信号を出力し、プロセススイッチを起こす手段を備える。

【0016】また、請求項4記載の発明では、スイッチ制御手段15は、前記スイッチ抑止状態におけるプロセススイッチの遅延が所定の時間を越えた場合には割り込みを起こし、一つのプロセスが長時間にわたって実行権を占有しないように制御する手段を備える。

【0017】さらにまた、請求項5記載の発明では、スイッチ制御手段15は、前記スイッチ抑止状態において非同期な外部割り込みがあった場合に、その割り込みの発生を遅延させ、現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベントが発生してから保留した割り込みを発生させる手段を備える。

【0018】請求項6記載の発明では、入出力を実行するプロセス、またはメッセージを送信もしくは受信するプロセスを一つに制限するために、一つのプロセスが入出力を開始またはメッセージの送信もしくは受信を開始してからそれが終了するまでのクリティカルな状態を管理し、そのクリティカルな状態における他の実行可能なプロセスの入出力起動またはメッセージ送信／受信の開始を抑止することにより、入出力またはメッセージの送信もしくは受信を、マルチプロセッシング環境においてもユーザプログラムのレベルで起動可能とする。

【0019】請求項7記載の発明では、入出力を実行するプロセス、またはメッセージを送信もしくは受信するプロセスを一つに制限するために、所定のアドレス空間のアクセス権を入出力を開始またはメッセージの送信もしくは受信を開始したプロセスに与え、他の実行可能なプロセスの前記アドレス空間へのアクセスを禁止し、前記アクセス権を持つプロセスの入出力が終了またはメッセージの送信もしくは受信が終了したときに前記アクセス権を変更することにより、入出力またはメッセージの送信もしくは受信を、マルチプロセッシング環境においてもユーザプログラムのレベルで起動可能とする。

【0020】

【作用】本発明では、例えば図1(B)に示すように、プロセススイッチ(タスクスイッチ)が起こるタイミングに幅を持たせることによって、プロセスが不可分な操作を行っている間に、プロセススイッチが発生することを防ぐので、マルチプロセッシング環境下においてもメッセージ送信／受信、入出力操作をユーザプログラムのレベルで実行することができるようになる。

【0021】プロセススイッチを起こすタイミングの周りで、メッセージ送信／受信または入出力の起動がかかった場合には、その起動を一時保留し、プロセススイッチを起こすようにするので、プロセススイッチの時間間隔が極端に長くなるのを事前に防ぐことが可能になる。

【0022】プロセスがメッセージ送信／受信または入出力中である場合には、プロセススイッチを起こすイベントを遅延させ、メッセージ送信／受信または入出力の終了イベントによってプロセススイッチを起こすことにより、送信／受信または入出力途中でプロセススイッチが起きないように制御することができる。

【0023】さらに、プロセススイッチを起こすイベントの遅延が、ある一定時間を越えた場合には、割り込みを起こすことにより、一つのプロセスがCPU時間を極端に長い時間占有しないようにすることができる。

【0024】非同期な割り込みに対しても、プロセスがメッセージ送信／受信または入出力中である場合には、その割り込みの発生を遅延させることにより、プロセスの不可分の処理を保証することが可能になる。

【0025】また、一つのプロセスが入出力またはメッセージ送信／受信を開始してから、終了するまでの間に、他のアクティブ(実行可能)なプロセスが入出力またはメッセージ送信／受信を開始できないように、プロセスが使用権を持つアドレス空間のアクセス権を制御することにより、入出力またはメッセージ送信／受信を行うプロセスを一つに制限し、マルチプロセッシング環境において、入出力またはメッセージ送信／受信を、複数のプロセスからユーザプログラムのレベルで起動できるようにすることが可能になる。

【0026】

【実施例】図2ないし図4は本発明の実施例における状態遷移説明図、図5および図6は本発明の実施例におけるプロセススイッチのタイミング説明図、図7は本発明の実施例による非同期割り込みの制御説明図、図8は本発明の実施例におけるアクセス権の制御説明図である。

【0027】請求項1記載の発明に対応する第1の実施例では、プロセススイッチ制御装置13は、図2(A)に示すように、free状態およびused状態の二つの状態を管理する。free状態は、現在実行中のプロセスが入出力中でもメッセージ送信／受信中でもなく、プロセススイッチを起こすことのできる状態である。used状態は、現在実行中のプロセスが入出力中またはメッセージ送信／受信中であり、プロセススイッチを起こすことを抑止しなければならない状態である。

【0028】初期化信号initial(またはリセット信号reset)のイベントにより、プロセススイッチ制御装置13はfree状態になる。free状態において、現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号startが入ると、許可信号ackを出力し、used状

態に遷移する。used状態においては、プロセススイッチを発生させるイベントが発生してもプロセススイッチを起こさずに、現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Endが入ったときに、許可信号ackを出力するとともに、プロセススイッチを起こすことを指示するスイッチ信号Switchを処理装置10に対して出力し、free状態に遷移する。

【0029】請求項2記載の発明に対応する第2の実施例では、プロセススイッチ制御装置13は、図2(B)に示すように、free状態、used状態およびswitch状態の三つの状態を管理する。free状態は、現在実行中のプロセスが入出力中でもメッセージ送信／受信中でもなく、プロセススイッチを起こすことのできる状態である。used状態は、現在実行中のプロセスが入出力中またはメッセージ送信／受信中であり、プロセススイッチを起こすことを抑止しなければならない状態である。switch状態は、プロセススイッチのトリガとなるタイマー信号timerが所定の時間内に発生する状態になったことを示すスイッチ予告促進状態である。

【0030】初期化信号initial(またはリセット信号reset)のイベントにより、プロセススイッチ制御装置13はfree状態になる。free状態において、現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Startが入ると、許可信号ackを出力し、used状態に遷移する。また、free状態において、プロセススイッチのトリガとなるタイマー信号timerが所定の時間内に発生することを示す予告タイマー信号timer0が入ると、switch状態に遷移する。

【0031】used状態においては、プロセススイッチを発生させるイベントが発生してもプロセススイッチを起こさずに、現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Endが入ったときに、許可信号ackを出力するとともに、プロセススイッチを起こすことを指示するスイッチ信号Switchを処理装置10に対して出力し、free状態に遷移する。

【0032】switch状態において、プロセススイッチのトリガとなるタイマー信号timerが入ると、スイッチ信号Switchを出力してプロセススイッチを起こし、free状態に遷移する。また、現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Startが入ると、許可信号ackを出力するとともに、スイッチ信号Switchを出力し、他の実行可能なプロセスに強制的にスイッチさせて、free状態に遷移する。また、switch状態でリセット信号resetが入ったときには、許可信号ackを出力し、free状態に遷移す

る

【0033】図5(A)は、この第2の実施例における動作例を示している。タスクAがfree状態で走行中に予告タイマー信号timer0が入ると、switch状態に遷移する。このとき、タスクAが入出力またはメッセージ送信／受信を開始しようとする、そのイベントを示す信号Startが検出されるが、switch状態になっているので、タスクスイッチを起こし、タスクAからCPU実行権を奪って他の実行可能なタスクBにCPU実行権を与える。

【0034】請求項3記載の発明に対応する第3の実施例では、プロセススイッチ制御装置13は、図2(C)に示すように、free状態、used状態、switch状態およびswitch0状態の四つの状態を管理する。free状態、used状態、switch状態は、前述した図2(B)に示す状態と同様である。switch0状態は、used状態において、予告タイマー信号timer0が入ったときに遷移するスイッチ準備状態である。

【0035】初期化信号initial(またはリセット信号reset)のイベントにより、プロセススイッチ制御装置13はfree状態になる。free状態において、現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Startが入ると、許可信号ackを出力し、used状態に遷移する。また、free状態において、プロセススイッチのトリガとなるタイマー信号timerが所定の時間内に発生することを示す予告タイマー信号timer0が入ると、switch状態に遷移する。

【0036】used状態において、現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Endが入ったときに、許可信号ackを出力し、free状態に遷移する。一方、used状態において、予告タイマー信号timer0が入ると、プロセススイッチの準備のためにswitch0状態に遷移する。

【0037】switch0状態では、現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Endが入ると、許可信号ackを出力するとともに、プロセススイッチを起こすことを指示するスイッチ信号Switchを処理装置10に対して出力し、free状態に遷移する

【0038】スイッチ予告促進状態であるswitch状態における状態遷移は、図2(B)で説明した状態遷移と同様である。図5(B)は、この第3の実施例における動作例を示している。タスクAが入出力またはメッセージ送信／受信を開始し、used状態になっているときに、予告タイマー信号timer0が入ると、switch0状態に遷移する。switch0状態になると、その後のプロセススイッチは遅延され、タスクAが

入出力またはメッセージ送信／受信を終了したことを示す信号Endによってタスクスイッチを起こし、タスクBにCPU実行権が与えられる。

【0039】請求項4記載の発明に対応する第4の実施例では、プロセススイッチ制御装置13は、図3(A)に示すように、free状態、used状態、switch状態およびswitch0状態の四つの状態を管理する。free状態、used状態、switch状態における状態遷移は、図2(C)に示す状態遷移と同様である。switch0状態についても、図2(C)の例とほぼ同様であるが、プロセススイッチを遅延させるスイッチ抑止状態(used状態またはswitch0状態)において、現在実行中のプロセスが長時間にわたってCPU時間を占有してしまうことを防ぐために、ある一定時間が経過した場合にはCPUに対して割り込み信号Interruptを出力してfree状態に戻す。

【0040】図6(A)は、この第4の実施例における動作例を示している。タスクAが入出力(I/O)を開始している場合には、タスクスイッチをある時間(TimeOut)だけ引き延ばし、入出力が終わるのを期待する。その時間内に終わらない場合には、その入出力を中断し、必要に応じてタスクAに関する処理途中の情報を保存して、タスクBにタスクスイッチする。

【0041】請求項5記載の発明に対応する第5の実施例では、プロセススイッチ制御装置13は、図3(B)に示すように、free状態、used状態およびmask状態の三つの状態を管理する。free状態、used状態については、前述の例と同様である。mask状態は、used状態において非同期な外部割り込みExInterruptがあった場合に遷移する状態である。非同期な外部割り込みExInterruptがあった場合にused状態からmask状態に遷移することにより、その非同期な外部割り込みの発生を遅延させ、ある一定の時間が経過したことを示すタイムアウトのイベント(TO: timeout)、あるいは現在実行中のプロセスの入出力またはメッセージ通信終了のイベントを示す信号Endによって、保留していた割り込みの信号Interruptを処理装置10に送出する。

【0042】図7は、この第5の実施例を実現するための非同期割り込み制御回路の例を示す。従来の通常の非同期割り込み等の割り込み信号は、OR回路60により論理的にORされ、CPUに対して与えられる。これに対し、本実施例では、例えば図7(B)に示すような制御回路が用いられる。現在実行中のタスクが入出力(またはメッセージ通信)を開始するときに、I/O Startの信号によってセット／リセットフリップフロップ61をセットし、入出力実行中であることを記憶しておく

この状態のときには、セット／リセットフリップフロップ61の出力であるmaskの反転信号は0であり、OR回路60の出力はAND回路62によってマスクされ、CPUに割り込みがかかるのが抑止される。これによって、現在実行中のタスクがクリティカルな入出力を行っている最中に、非同期割り込み等が発生するのを防ぐことができる。実行中の入出力が終了すると、I/O End信号によってセット／フリップフロップ61がリセットされ、割り込みのマスクは解除される。

10 【0043】請求項6記載の発明に対応する第6の実施例では、プロセススイッチ制御装置13は、図4(A)に示すように、free状態、used状態の二つの状態を管理する。本実施例では、例えば一つのプロセスがメッセージを送出し始めてから、送出し終わるまでの間に、他のアクティブ(実行可能)なプロセスがメッセージの送出を行えないようにするために、次のように制御する。

【0044】初期化信号initial(またはリセットreset)のイベントにより、プロセススイッチ制御装置13はfree状態になる。free状態において、現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Startが入ると、許可信号ackを出力し、used状態に遷移する。used状態においては、他のアクティブ(実行可能)なプロセスが入出力の起動またはメッセージ通信の開始を指示し、入出力またはメッセージ送信／受信を開始するというイベントを示す信号Startを出力しても、不許可信号nackが返される。そして、現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Endが入ったときに、許可信号ackを出力するとともに、プロセススイッチを起こすことを指示するスイッチ信号Switchを処理装置10に対して出力し、free状態に遷移する。

【0045】図6(B)は、この第6の実施例における動作例を示している。タスクA、タスクB、タスクCが走行可能状態であったとする。本実施例は、例えばタスクAが入出力(I/O)処理を開始した状態で、タスクBやタスクCにCPU実行権が切り替わった場合には、タスクBやタスクCに対してI/Oにアクセスする権利を与えないようにするものである。図6(B)に示す動作例では、タスクAが入出力処理を終えた後に、タスクBが入出力処理を開始している。

50 【0046】請求項7記載の発明に対応する第7の実施例では、例えば図8(A)に示すように、タスクA、タスクB、タスクCが走行可能状態にある場合に、入出力(I/O)を起動できるアドレス空間(User accessible I/O)を、タスクAのみに解放して、他のタスクB、Cからはこのアドレス空間にアクセスできないようにアクセス権を制限する。アクセス権の制限は、例えば主記憶

保護などの周知の技術を用いて実現することができる。

【0047】さらに、この第7の実施例において、アクセス権が制限されているタスクB、CがI/Oをアクセスしようとする、例えば割り込みを発生させてOSに制御を移し、OSは割り込みを起こしたタスクBまたはタスクCの状態を変化させ（I/O待ちなど）、タスクAのI/Oが終了するまで、I/O待ちなどの状態にあるタスクB、CにCPU時間を与えないようにする。図4（B）はその例における状態遷移を示している。タスクAのI/Oの開始により、free状態からused状態に遷移する。このused状態において、他のタスクB、CがI/Oを起動しようとしても、タスクスイッチによってI/O待ちになる。タスクAのI/Oが終了するとfree状態に戻り、他のタスクB、CのI/Oの起動が可能になる。

【0048】図7（B）は、この第7の実施例を実現するためのシステム構成の例を示している。図中、50はCPU、51はメモリ制御装置（MMU）、52はメモリを表す。メモリ制御装置51は、CPU50が出すアドレスをチェックする機構を持つ。OSは、どのタスクがI/O領域にアクセスできるかをメモリ制御装置51のモジュールに設定しておく。これにより、アクセス権のないタスクが発行したアドレスが不適切である場合には、割り込み（Interrupt）を発生させ、タスクスイッチを起こす。メモリ制御装置51によるアクセス保護機能の利用はマルチタスクの環境では一般的であり、本実施例を実現するための装置の変更はほとんど必要がない。もちろん、これと同等な機能を持つアクセスチェック機構によって本実施例を実現することも可能である。

【0049】図9（A）は、本発明の実施例によるプロセススイッチ制御装置の構成例を示す図であり、図9

（B）はプロセススイッチ制御装置におけるカウンタの出力説明図である。また、図10および図11は、プロセススイッチ制御装置におけるステートマシンの実現方法説明図である。

【0050】以下では、請求項3記載の発明の一実施例であるプロセス制御装置の構成例について説明する。図9（A）において、90はステートマシン（FSM）であって、有限な状態を管理し入力信号によって状態を遷移させるとともに、そのときの状態と入力信号とに応じて各種の制御信号を出力する装置である。91はCPUが出すアドレスをデコードするデコーダ（DEC）、92はCPUのクロックclkをカウントし、所定の値になったときに予告タイマー信号timer0を出力する第1のカウンタ（counter0）、93はクロックclkをカウントし、所定の値になったときにプロセススイッチのトリガとするタイマー信号timerを出力する第2のカウンタ（counter）を表す。

【0051】CPUが起動するI/Oやメッセージ送出

命令は、この例では、CPUが出すアドレスによって検出されるデコーダ91は、そのようなアドレスをデコードすることによって、CPUのクリティカルな操作の開始時（I/Oの開始、メッセージ送出の開始等）に信号Startを生成し、クリティカルな操作の終了時に信号Endを生成する。同様に、リセット信号resetもCPUが出す特定のアドレスによって生成する。

【0052】また、第1のカウンタ92および第2のカウンタ93は、CPUのクロックclkを分周し設定された時間に、それぞれ予告タイマー信号timer0およびタイマー信号timerをアクティブにする。このカウンタ出力信号のタイミングは、例えば図9（B）に示すようになっており、タイマー信号timerがアクティブになる所定の時間前に、予告タイマー信号timer0がアクティブになるように設定されている。タイマー信号timerがアクティブになる時間間隔は、通常の場合におけるプロセススイッチの時間間隔に一致し、この時間はCPUのクロックclkに対して十分に長い時間である。

【0053】これらの信号はステートマシン90に入力され、ステートマシン90によって、例えば図2（C）に示すような状態遷移に応じてack（nack）、Switchの信号が生成される。CPUは許可信号ackを受け取ると正常にバストランザクションが終了（成功）したことを認識する。許可信号ackが出力されなかった（または不許可信号nackが出力された）場合、CPUはアクセスが正常に終了していないことを認識して、リトライを行うか停止する。また、スイッチ信号Switchによって、他のプロセスを実行するためのプロセススイッチを起こす。

【0054】以上のようなステートマシン90は、図10および図11に示すようなハードウェア記述言語をコンパイルすることによって、容易に実現することができる。図10および図11は、ハードウェア記述言語の一つとして周知のVerilogを用いて第4の実施例における状態遷移を実現するものを示している。Verilogはよく知られた言語であるので詳しい説明を省略し、ここでは図10および図11による表現の概要だけを簡単に説明する。

【0055】第2行目は、ステートマシンのモジュールの定義文である。第3行目は、入力信号を示す。ここでintは、図2においてinitialとして示されている初期化信号である。第4行目は、出力信号を示す。

【0056】第6行目は、現在の状態cstateと次の状態nstateを管理するレジスタを定義している。第8行目は、2ビットでfree状態（FREE＝“00”）、used状態（USED＝“01”）、switch0状態（SWITCH0＝“10”）、switch状態（SWITCH＝“11”）の状態を表すことを定義している。これらの状態管理は2個のフリ

ップフロップで実現される。

【0057】第9行目から第38行目は、cstate, end, start, reset, timer, timer0またはintにより、次の状態がどのように遷移するかを示している。

【0058】また、第40行目から最終行までは、ackおよびswitchの信号を出力する条件を示している。ここでは、第4の実施例を代表させて、図9(A)に示すステートマシン90を実現するVerilog記述の例を示したが、他の実施例におけるステートマシンの状態遷移についても、以上説明した例から推測して当業者であれば容易に実現可能であるので、個々の実現例のこれ以上の詳しい説明は省略する。図10および図11に示すVerilog記述をコンパイルすることによって、図9(A)に示すステートマシン90を実現する論理回路を自動設計する技術は確立されている。

【0059】

【発明の効果】以上説明したように、本発明によれば、ユーザレベルでのメッセージの送出や、I/Oの実現におけるプロセススイッチに関する問題を解決することによって、マルチプロセッシング環境下でも、ユーザレベルで直接メッセージの送出やI/Oを制御することができ、低いオーバーヘッドでこれらを実現することが可能になる。したがって、計算機の性能を効率よく引き出すことが可能となる。特に、プロセッサ間のメッセージ通信が頻繁に現れるような並列計算機システム等においては性能向上の効果が大きい。

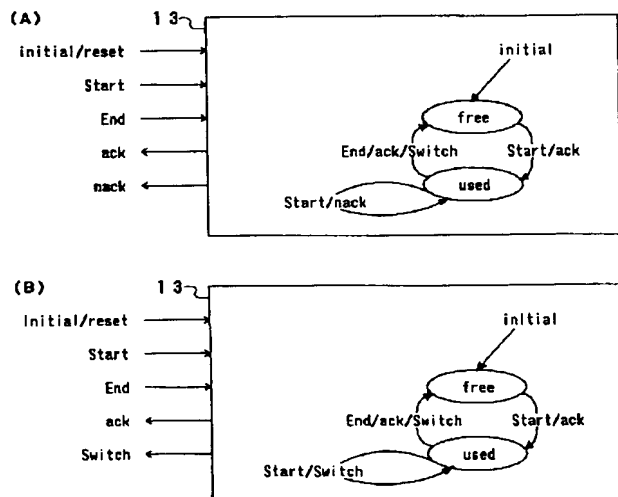
【図面の簡単な説明】

【図1】本発明の原理説明図である。

【図2】本発明の実施例における状態遷移説明図であ

【図4】

状態遷移説明図



る。

【図3】本発明の実施例における状態遷移説明図である。

【図4】本発明の実施例における状態遷移説明図である。

【図5】本発明の実施例におけるプロセススイッチのタイミング説明図である。

【図6】本発明の実施例におけるプロセススイッチのタイミング説明図である。

【図7】本発明の実施例による非同期割り込みの制御説明図である。

【図8】本発明の実施例におけるアクセス権の制御説明図である。

【図9】本発明の実施例によるプロセススイッチ制御装置の構成例を示す図である。

【図10】本発明の実施例によるプロセススイッチ制御装置におけるステートマシンの実現方法説明図である。

【図11】本発明の実施例によるプロセススイッチ制御装置におけるステートマシンの実現方法説明図である。

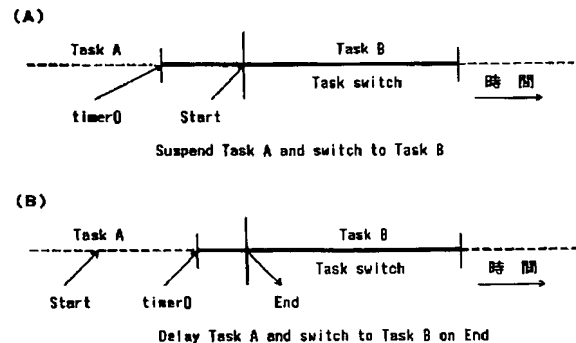
【図12】本発明の課題説明図である。

【符号の説明】

- 10 処理装置
- 11 オペレーティング・システム(OS)
- 12A, 12B プロセス
- 13 プロセススイッチ制御装置
- 14 状態管理手段
- 15 スイッチ制御手段
- 16 転送制御装置
- 17 I/Oデバイス

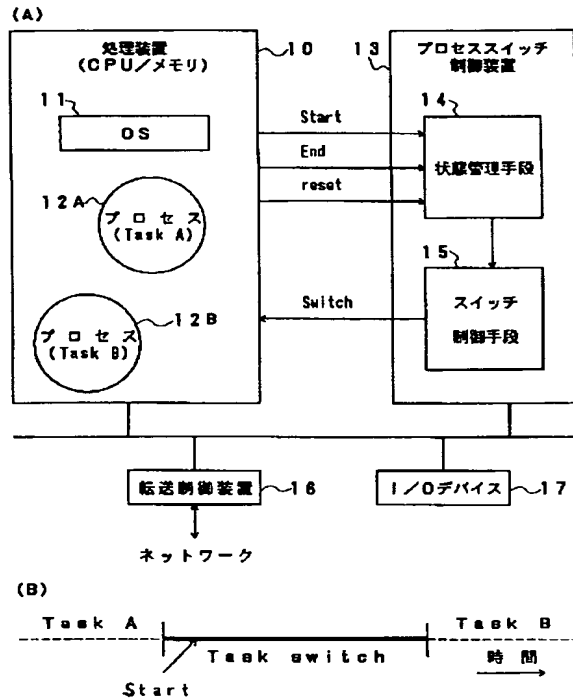
【図5】

プロセススイッチのタイミング説明図



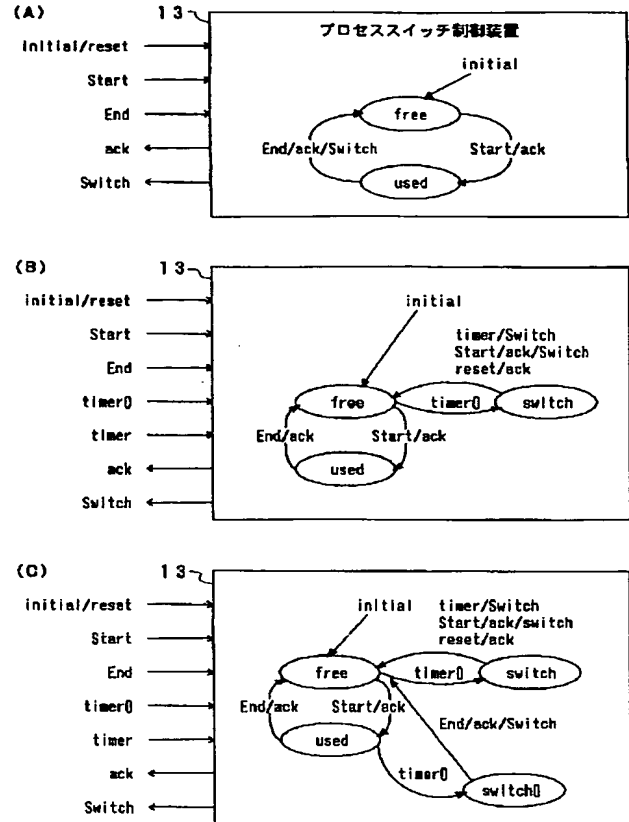
【図1】

本発明の原理説明図



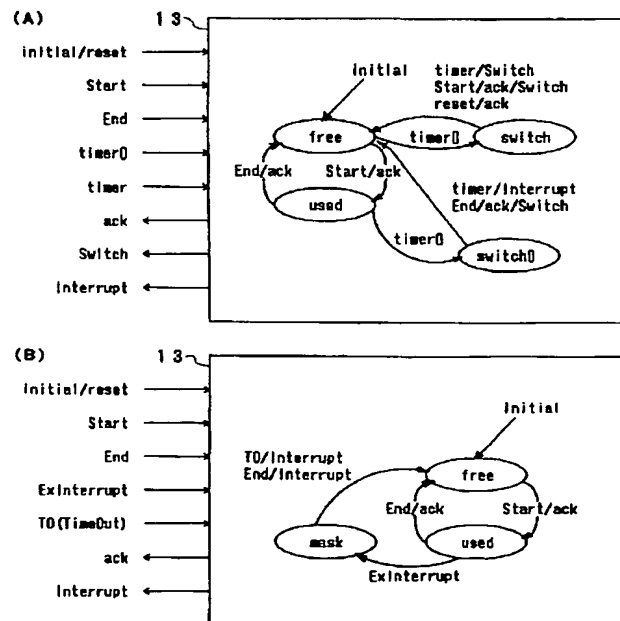
【図2】

状態遷移説明図



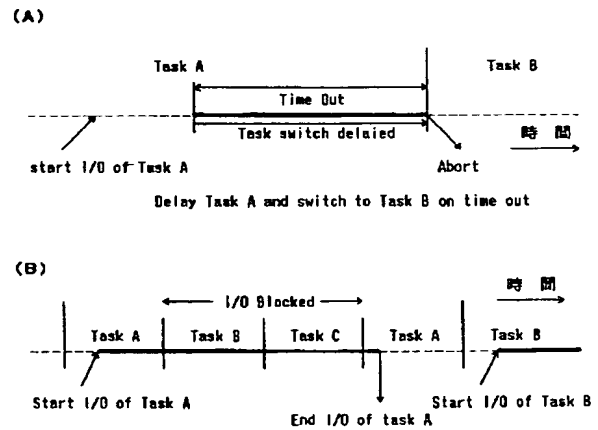
【図3】

状態遷移説明図



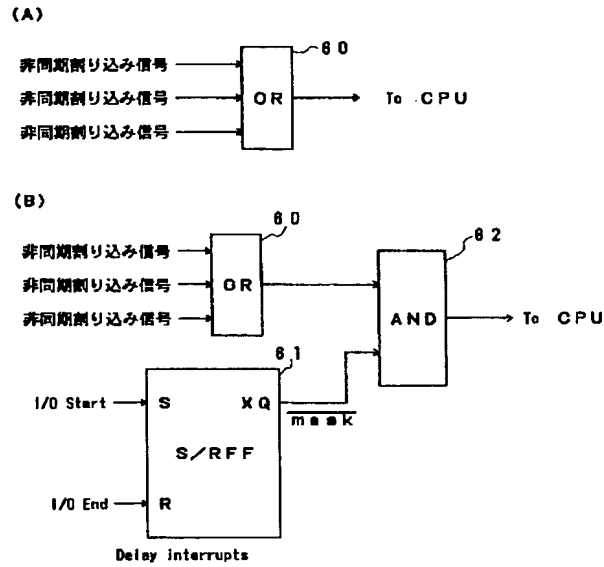
【図6】

プロセススイッチのタイミング説明図



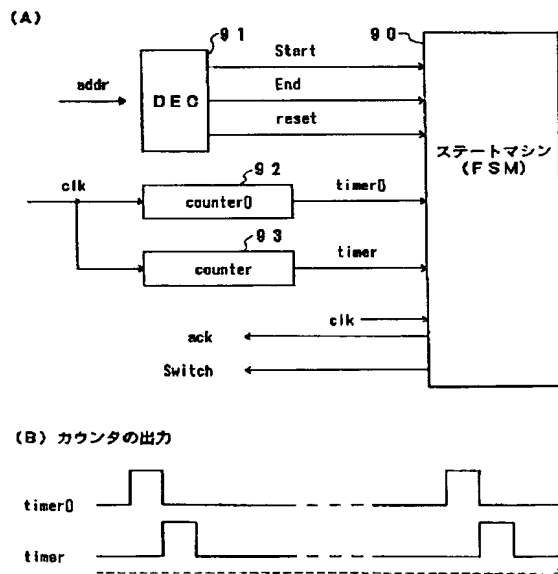
【図 7】

非同期割り込みの制御説明図



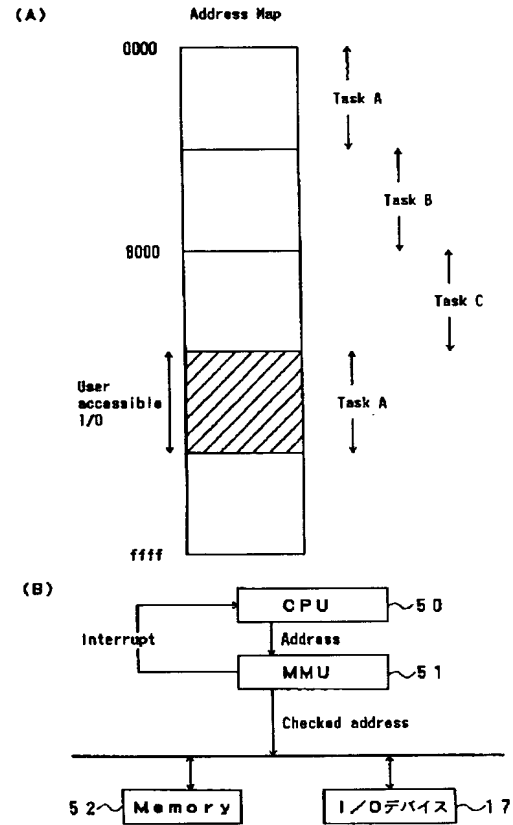
【図 9】

プロセススイッチ制御装置の構成例



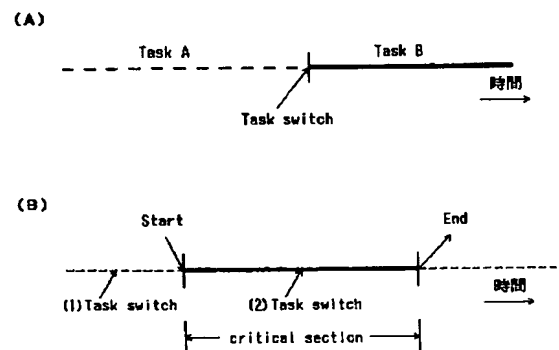
【図 8】

アクセス権の制御説明図



【図 12】

本発明の課題説明図



【図10】

ステートマシンの実現方法

```

1 module fsm(end,start,reset,timer,timer0,clk,int,ack,switch);
2 input end,start,reset,timer,timer0,int,clk;
3 output ack,switch;
4
5 reg [1:0] cstate,nstate;
6
7 parameter [1:0] FREE = 2'b00, USED = 2'b01, SWITCHO = 2'b10, SWITCH = 2'b11;
8
9 always @ (cstate or end or start
10   or reset or timer or timer0 or int)
11 begin
12   nstate = cstate;
13   if (int)
14     nstate = FREE;
15   else case (cstate)
16     FREE:
17       if (start)
18         nstate = USED;
19       else if (timer0)
20         nstate = SWITCH;
21   endcase
22 end
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```

【図11】

ステートマシンの実現方法

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```

```

USED:
  if (end)
    nstate = FREE;
  else if (timer0)
    nstate = SWITCHO;
  SWITCHO:
    if (end || timer)
      nstate = FREE;
  SWITCH:
    if (reset || start || timer)
      nstate = FREE;
  endcase
end
always @ (posedge clk)
begin
  cstate = nstate;
end
wire ack = {cstate == FREE} && start
           || {cstate == USED} && end
           || {cstate == SWITCHO} && end;
           || {cstate == SWITCH} && (reset || start);
wire switch = (cstate == SWITCH) && (start || timer)
              || (cstate == SWITCHO) && end;
endmodule

```

This Page Blank (uspto)